面向 MapReduce 计算的大规模集群通信优化 *

曹云鹏 [1,2]，王海峰 [1,2†]，刘海涛 [1,2]，何淑庆 [1,2]

(1. 临沂大学 信息科学与工程学院，山东 临沂 276002；2. 山东省网络重点实验室临沂大学研究所，山东 临沂 276002)

摘　要：为了优化大规模集群运行 MapReduce 作业时的通信效率和减少 Shuffle 数据传输量。首先采用存储局部性换取通信局部性的策略，建立一个分布式协同数据映射模型；其次通过随机抽样和机器学习方法来提取作业数据的局部性特征，实现 map 计算数据的有效部署；最后，利用软件定义网络的全局灵活控制能力，优选通信链路好的节点并将计算任务映射到该类节点中。实验表明对于中间数据混洗密集类作业有较好的优化效果，通信延迟降低了 4.3%~5.8%。该方案能减少 Shuffle 流量和数据迁移延迟，并且适合各种调度策略和网络拓扑结构。

关键词：数据通信优化；MapReduce；软件定义网络；协同数据映射

中图分类号：TN915.02　　doi: 10.19734/j.issn.1001-3695.2018.10.0766

Communication optimization for large-scale cluster aiming MapReduce computing

Cao Yunpeng[1, 2], Wang Haifeng[1, 2†], Liu Haitao[1, 2], He Shuqing[1, 2]

(1. *School of Information Science & Engineering*, *LinYi University*, *Linyi* Shandong 276002, *China*; 2. *Linda Institute*, *Shandong Provincial Key laboratory of Network Based IntelligentComputing*, *Linyi* Shandong 276002, *China*)

Abstract: To optimize communication efficiency and reduce the data transmission of shuffle in large-scale clusters running for MapReduce jobs, a distributed collaborative data mapping model was built based on the strategy, which is designed by replacing the communication locality by storing locality. Then the local features of jobs are extracted by random sampling and machine learning method in order to realize the effective deployment of Map tasks. Finally, the good communication links are selected based on the software define network technology due to with the global flexible control capabilities. And the Map Tasks are scheduled to such nodes. Experimental results show that it has better optimization effect on shuffle-intensive jobs. The communication delay is reduced by 4.3% to 5.8%. This solution can reduce Shuffle traffic and data migration delay. It is suitable for various scheduling strategies and network topologies.

Key words: data communication optimization; MapReduce; software-defined network; collaborative data Mapping

0　引言

随着大数据分析和实时计算的普及，大规模集群中运行着大量 MapReduce 模式的作业。MapReduce 把计算分成 Map 和 Reduce 两个阶段，在 Map 中输入数据分成多个数据块并被映射各节点中完成并行计算。Map 计算的中间结果需要通过数据混洗实现重定位。在数据混洗过程中相同 Key 值的数据要从各节点汇聚到同一节点中来完成 Reduce 计算。在数据混洗中会出现多个节点向同一节点传输数据的 incast 问题。目前数据中心中数据混洗的网络传输量占到整个网络流量的 58.6%[1]，中间数据通信成为 MapReduce 计算的一个重要性能瓶颈，降低中间数据传输延迟是提高大数据计算性能的迫切需要。

目前，对于 MapReduce 计算模式中间数据传输优化的研究，主要分为四个方面：

a)粗粒度的数据块均衡映射。在 Hadoop 中数据分块的映射采用简单的 Hash 函数法，导致数据分块分布不平衡。为了解决该问题 Ibrahim 等提出感知公平的数据键值分区方法，根据数据键值频率分布来设计数据块映射方案[2]; Palanisamy 设计资源优化分配系统来提高 MapReduce 的计算性能，将中间数据分配到具有存储局部性的位置来减少数据传输量[3]；在 iShuffle 系统中分离数据混洗与 Reduce 过程。在抽取的 Shuffle 服务中重新映射数据块来均衡中间数据倾斜和降低 Shuffle 的写延迟[4]。

b)细粒度的中间数据汇聚优化研究。采用网内关联性流量汇聚后再传输的方式，降低中间数据传输延迟[5]；Ke 等在 Reduce 阶段前增加一个数据汇聚层，将中间数据汇聚转换成混合整数非线性规划问题，再设计分布式数据分发算法来优化中间数据传输[6]。

c)粗粒度的优化中间数据的放置研究。将中间数据放入 RDMA 中间存储层中，以此提高中间数据读写效率和减少网络中数据传输量[7]；提出一种新的判断相似性的距离模型，重新计算数据中心节点之间的网络距离，尽量将 Reducer 任务放置到相似的节点中。通过计算靠近数据的方法来减少 Shuffle 数据量[8]。

d)从优化数据中心网络体系角度来研究[9]。针对数据中心胖树网络结构提出优化核心网络流量的方法，来解决 Shuffle 通信流量产生的网络拥塞问题[10]。软件定义网络 SDN

收稿日期：2018-10-23；**修回日期：**2018-12-14　**基金项目：**山东省自然科学基金面上项目（ZR2017MF050）；山东省高等学校科学技术计划项目（J17KA049）；山东省重点研发项目（2018GGX101005, 2017CXGC0701, 2016GGX109001）；山东省自主创新及成果转化专项资助项目（2014ZZCX02702）

作者简介：曹云鹏（1967-），男，副教授，硕士，主要研究方向为大数据并行计算；王海峰（1976-），男（通信作者），教授，硕导，博士，主要研究方向为高性能计算、大数据分析（gadfly7@126.com）；刘海涛（1977-），男，讲师，博士研究生，主要研究方向为软件定义网络；何淑庆（1982-），讲师，博士，主要研究方向为分布式计算，大数据分析.

作为一种逐渐成熟的新技术也成为一种优化 Shuffle 流量的方法。利用 SDN 来优化计算集群的网络链路，优化计算节点之间链路质量来减少数据传输延迟[11]；利用 SDN 能掌握网络全局信息的能力设计一种感知带宽的任务调度模型，在数据中心网络范围内保证了数据的局部性，提高计算作业的性能[12]；利用 SDN 对网络灵活的控制能力，在 OpenFlow 交换机上完成 Shuffle 中间数据合并，减少数据流量和传输时间[13]；在 SDN 基础上提出一个针对 Shuffle 流量优化的路由算法，根据 Shuffle 流的特征选择高效的转发路径，该方案不仅能提高计算性能而且能增加网络稳定性[14]。综上所述，在 Shuffle 阶段实现细粒度数据汇聚来减少通信量，虽然准确性较高，但是算法复杂性高。本文采用粗粒度的数据块均衡映射方案来提高 Shuffle 阶段的数据局部性，并且利用 SDN 能够灵活掌握网络全局链路的特性把数据块提前调度到通信链路质量高的节点中。本文结合数据局部化来优化 MapReduce 作业调度算法，降低中间数据传输量和通信延迟。具有两个优点：由于数据映射优化工作在真正调度之前执行，因此适合任何 MapReduce 调度算法的改进；在 SDN 灵活控制网络全局的支持下优化集群通信链路，不受网络拓扑结构的限制。

1　问题描述

在 MapReduce 模型中，分布在各节点的 Map 任务要将中间数据汇聚起来作为 Reduce 任务的输入，这就是中间数据混洗的 Shuffle 阶段。在 Shuffle 阶段会出现大量节点向同一节点传输数据的现象，因此容易引起网络性能瓶颈。为了提高网络性能提出一种利用数据存储局部性来优化网络通信的协同数据映射模型。协同数据映射模型是计算任务的输入数据集和计算节点集之间的函数映射，利用该映射实现输入大数据合理有效的分割与部署。

设计算任务 T_i 的输入数据集 $D_i=\{B_i, i=1...m\}$，集群为计算节点集 $\{N_j, j=1...n\}$，则函数 f 表示协同数据映射模型。

$$f:\{B_i\}\rightarrow\{N_j\}(i=1..m,\ j=1..n) \tag{1}$$

如图 1 所示，计算作业输入数据 GA 以一个全局数组形式存在，该全局数组存放数据存储的元数据，即每个数据分块的分布情况。数据分块（Data Block）是一个向分布式存储系统直接映射的物理存储单元，可分布在各个节点的磁盘。比如，GA=（B_1,B_2,B_3,B_4,B_5,B_6），存放到 4 个计算节点 [$Node_1,Node_4$] 中，可表示为 GA=（$B_1(N_1),B_2(N_1),B_3(N_2),B_4(N_3),B_5(N_4),B_6(N_4)$），并且满足式(2)的约束。

$$Min(\sum Cost(N_{ij}))\ (i\neq j) \tag{2}$$

式(2)中计算任务的数据映射到一个计算节点集合中，该集合各节点之间网络通信代价最小。总之，要解决的问题是实现映射节点之间数据传输代价最优问题。

2　协同数据映射模型

协同数据模型要解决计算数据分布的辨识问题。数据分布辨识的方案如下：设计算作业的数据由各数据子集 $\{B_1,B_2,...,B_i,B_j,...B_n\}$ 构成，Cost 为通信代价函数（该函数包括节点内通信、节点间通信,其中节点间通信分成机架内和跨机架通信代价），数据布局时的限制条件如式（3）所示。

$$Min(\sum Cost(B_i\cap B_j))\qquad (i\neq j,\ i=1..n,\ j=1..n) \tag{3}$$

在满足该限制条件下辨识数据映射函数关系。根据基本的函数映射关系，采用求解反函数的思路。映射函数的输入是合理的数据块布局，输出值为具有通信高效的节点子集。根据反函数求解先确定数据布局映射函数的值域，即明确具有通信局部性的计算节点，比如位于同一机架 R_1 中的节点集合 $\{N_1,N_2,...N_k\}$。在函数的因变量确定后，寻找相应函数自变量的值 $\{B_1,B_2,...,B_t\}$。在 MapReduce 计算模式中，计算任务 T_i 被划分成子任务 $T_i=\{Ts_1, Ts_2,...Ts_t\}$，每个子任务并发处理输入数据的子集 $D_i=\{B_1,B_2,...,B_t\}$。采用式（4）实现数据映射函数的决策空间转换：

$$f_1:\{B_i\}\rightarrow\{N_j\}\Rightarrow f_2:\{Ts_i\}\rightarrow\{N_j\} \tag{4}$$

经过决策空间转换后，数据映射问题变成寻找具有通信局部性的计算子任务问题。因此输入数据的决策空间转换到计算任务的特征空间中了，再从计算任务的特征空间中寻找决策属性，将计算任务划分成通信局部性好和差的两大类。然后把通信局部性较好的节点集映射到网络链路好的节点中，比如位于同一机架交换机内的物理节点。

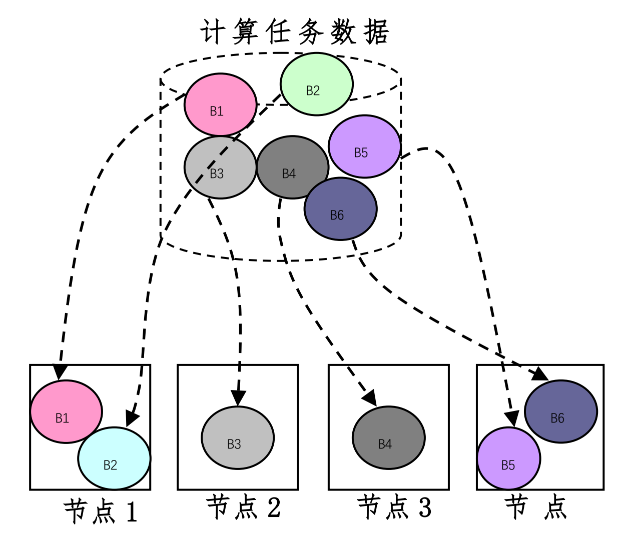


图 1 数据映射示意图

Fig. 1　Data mapping schematic diagram

3　通信优化方案

3.1　作业分类预测

计算作业分类预测的主要思想是在异构集群中运行 MapReduce 作业，提前计算作业的特征，并且记录各样本作业的网络通信过程，量化数据传输量并且建立分类指标。通过机器学习的方法建立分类器，训练分类模型后实现在线预测。根据该思路要解决计算作业特征提取、分类指标量化和作业分类器选择三个问题。

3.1.1 计算作业特征

选择 MapReduce 计算作业的特征是建立分类模型训练的前提条件，然而无法提取 MapReduce 作业的运行时性能参数[15]，比如 Map,Reduce 的执行时间以及各节点集群硬件资源的利用率。这些运行信息需要作业执行后才能获得，无法应用到静态的作业分类预测中。因此选择计算作业部署运行前的信息作为样本特征，具体如下：

MapReduce 作业提交计算平台之前，用户对计算作业提供两个基本信息：

a)作业的基本类型 T。MapReduce 作业的基本类型有 CPU 密集型作业、I/O 及网络密集型作业、CPU&I/O 混合型作业三种。采用高、中和低三个模糊等级来衡量三种类型，例如：CPU 密集型作业{高=0.1，中=0.2，低=0.3}、CPU&I/O 混合型作业{高=0.6，中=0.5，低=0.4}和 I/O 及网络密集型作

业{高=0.9，中=0.8，低=0.7}。

b)计算作业的输入数据规模 S。输入数据规模也采用高、中和低三个模糊等级来量化。主要是根据专家的经验值确定数据规模的大小范围，再界定高、中和低，分别对应 0.9,0.6 和 0.3。

另一方面，MapReduce 计算作业由调度程序预先部署到集群中。在此提取作业执行前的部署信息来作为分类特征。设集群包括若干机架 $G=\{R_1,R_2,...,R_p\}$，每个机架中有计算节点 $R_i=\{n_1,n_2,...,n_q\}$。调度程序根据 Map 任务处理的数据块来实现 Map 任务的划分，如式(5)所示。

$$J_i=\begin{vmatrix} 1 & 0 & 1 & 3 & 5 & 0 & 3 \\ 8 & 1 & 0 & 4 & 0 & 0 & 6 \\ 0 & 0 & 1 & 5 & 0 & 7 & 5 \\ 2 & 0 & 2 & 0 & Null & 7 & 6 \\ 0 & 6 & 0 & 4 & 0 & 2 & 9 \\ 4 & 5 & 0 & 3 & 1 & 1 & 13 \end{vmatrix} \tag{5}$$

该行列式的行表示机架，列表示机架内的节点。这里共 6 个机架，每个机架有 7 个节点。作业 J 的 Map 任务分布为一个任务矩阵 $J=[T_{ij}]_{p\times q}$，T_{ij} 表示映射到该节点 n_{ij} 中的 Map 任务数量；若集群中的某节点失效或者不存在，则设置 T_{ij} 为空(Null)。例如 $T_{0,4}=5$ 表示 R_0 机架中第五个节点 $n_{0,4}$ 中分配了 5 个任务；所有零元素代表该节点未分配到任务，比如 $T_{0,1}=0$；Null 节点表示该节点不存在或者处于故障状态。在作业 Map 任务矩阵基础上引入两个特征指标：

a）Map 任务分布稀疏度。设集群中计算节点总数为 n，被划分任务的节点数为 m，则分布稀疏度 γ 如式(6)所示。

$$\gamma=m/n\ (n>0,m>0,m<n) \tag{6}$$

分布式稀疏度刻画了 Map 任务划分的广度，该值越大说明任务划分越细，涉及的计算节点越多，作业计算的中间数据传输越复杂。

b)Map 任务分散度。设集群中机架数为 p，被分配任务的节点所占据的机架数为 k,则分散度 λ 如式（7）所示。

$$\lambda=k/p\ (k>0,p>0,k\le p) \tag{7}$$

分散度描述 Map 任务跨机架划分的程度，该值越大则说明跨机架数据传输越复杂。

3.1.2 通信活跃度

本节将作业通信活跃度指标用来分类，把计算作业分成通信活跃和不活跃两大类。选择 MapReduce 作业的数据 Shuffle 阶段跨机架数据传输作为观测数据，设作业 $J_i=\{n_1,n_2,...,n_p\}$，节点之间相互通信，因此 J_i 也是一个通信集合，在该通信集合中跨机架节点之间的数据通信量表示为 d_{ij}，

$$d_{ij}(<n_i,n_j>,n_i\in R_i,n_j\in R_j,i\ne j) \tag{8}$$

其中：R 为机架，J_i 的数据混洗阶段的通信量为 D_i:

$$D_i=\sum_{j=1}^{k}d_j \tag{9}$$

式（9）中 k 是跨机架通信的节点对数目。在此将作业 J_i 的通信活跃度 CA_j（communication activity）定义为该作业涉及到的跨机架数据通信量的平均值如下：

$$CA_i=D_i/k\times D_{max} \tag{10}$$

式（10）中 D_{max} 为单一节点对最大通信量，把 CA_j 归一化到[0,1]中。接着要确定作业分类的阈值，通过通信活跃度阈值实现计算作业的分类。在此用实验分析样本的方法来辨识阈值。先在 MapReduce 基准程序集和现有大数据分析作业中选 50 个样本作业，再监控数据 Shuffle 阶段中的网络流量，并统计跨机架通信量；最后用式（10）计算各作业的通信活跃度，样本作业的通信活跃度的分布如图 2 所示。

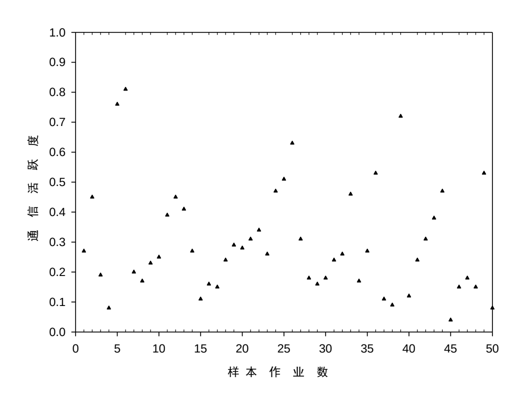


图 2　样本通信活跃度分布

Fig. 2　Sample communication activity distribution

通过观察样本作业通信活跃度的分布可看出，在 0.3~0.4 存在一个明显的分界线。利用样本通信活跃度置信区间分析，接受 30%的作业作为通信活跃作业时阈值可设定为 0.38。

3.1.3 分类预测器

计算作业类型预测模型是一个分类模型，输入作业特征向量<T,S, γ, λ >,预测输出为通信活跃度 CA，根据通信活跃度的阈值分成两大类。在此采用 BP 神经网络作为分类器，BP 神经网络是一个利用反向传播来调整网络参数的传统模型，具有简单和便于实现的优点[16]。BP 神经网络拓扑结构采用典型的三层结构：输入层、隐层和输出层。各层神经元节点分别为 4,6,1，隐层神经元个数通过经验值设定。隐层神经元传递函数为 $g(x)=1/1+e^{-x}$,网络学习率 0.3,样本容量为 100,网络经过 562 次迭代后收敛。

3.2　网络链路优化设计

本节寻找集群中通信局部性好的节点子集，这些节点之间传输链路质量高。通过将 MapReduce 任务部署到通信局部性较好的节点中，以此减少多个计算任务 Shuffle 过程的数据通信量，提高集群的通信效率。在传统集群中通信局部性较好的节点严重依赖网络拓扑结构。比如在 Fat-Tree 结构中位于同一个 Pod 内的节点子集具有较好的通信局部性，所有与机架内交换机连接的节点具有较高的通信效率。然而这种依赖集群网络拓扑结构的方案存在严重缺点：a)扩展性差。当集群中出现增删节点或者节点故障离线时就影响通信优化方案，需要及时掌握网络拓扑变化；b）缺乏动态部署能力。当用户作业负载出现变化时，无法及时掌握网络全局信息，因此缺乏动态调整网络拓扑结构的能力。

为了解决传统集群网络结构的问题采用软件定义网络(software define network,SDN)结构。SDN 是一种新兴的网络架构，其核心思想是分离网络硬件设备的数据转发和决策控制功能，实现硬件实现数据转发和软件控制转发的逻辑决策[17]。SDN 主要分成应用层、控制层和数据层，如图 3 所示。控制层是由基于软件的 SDN 控制器组成，控制器负责维护整个网络全局视图，提供集中式的控制功能；数据层包括物理或者虚拟交换机等数据转发单元，根据控制层下发的转发规则实现网络数据的转发功能；应用层主要是各类网络服务的应用构成，而 MapReduce 计算也是作为一种应用可以在 SDN 架构支持的应用层中运行。

下面详细介绍在 SDN 架构的集群网络中如何获得通信局部性较好节点集的方案。本文的目的是为了找到计算节点子集{$N_1,N_2,...N_m$}，该集合的网络链路为{$e_1,e_2,...e_k$}通信效率最优。因此求通信局部最优节点集转换成寻找集群网络中最优链路子集的问题。在 SDN 网络中通过测量链路可用带宽并排序来获得最优链路子集。设在时间间隔 $(t-\mu,t)$ 内，通过链

路 e_i 的流量为 l_i,e_i 的传输总容量为 c_i,则可用带宽 σ_i 为

$$\sigma_i(t-\mu,t)=c_i\times l_i(t-\mu,t)/\mu \qquad (11)$$

该求解最优链路子集的功能模块放置到 SDN 的控制器内，由于 SDN 控制器从全局范围掌握网络的拓扑和流量信息，采用分布式方式管理各个 OpenFlow 交换机(Open vSwitch,OVS)。SDN 控制器通过 OpenFlow 协议发出交换机状态的查询指令，OVS 交换机的 Local 端口接受到 SDN 的控制消息 OFQueueStatsRequest 后，会产生一个响应消息 OFQueueStatsReply。在该消息有个 tx_byte 字段统计传输字节数，查询消息的时间粒度是纳秒级。然后发起连续时间间隔内的两次请求就能统计出该条链路的可用带宽，这里设置的间隔是 1s。最后遍历集群网络中的链路集合后可求出最优的链路子集，以及最优链路子集所对应的计算节点集合。

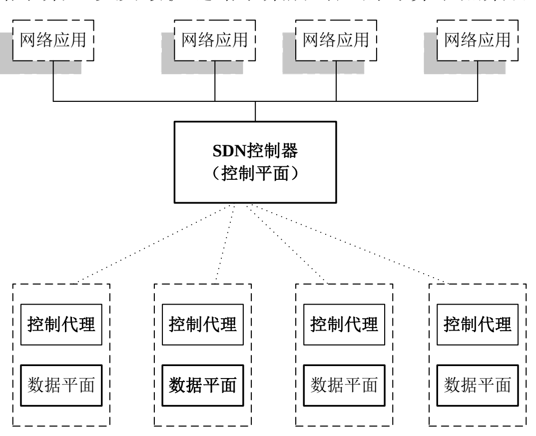


图 3　SDN 架构示意图

Fig. 3　SDN architecture schematic diagram

3.3　通信能效优化部署算法

针对 MapReduce 计算模式 Shuffle 阶段的大量数据通信过程，根据作业类型预测结果来重新调度任务的分布。基于 SDN 架构实现集群中的网络互连，SDN 控制器端实时获取网络拓扑结构和链路等信息。将通信活跃的作业重新部署到最优链路集合的节点中；对于通信惰性的作业则维持原部署方案。该算法核心思想是尽量控制任务之间的通信局部性，减少集群中数据在质量差链路中的性能损失，并以此提高中间数据通信效率。具体算法如下所示：

算法 1：通信优化部署算法

输入：大数据作业队列 Q(J1,J2,...,Jn)，计算作业预部署信息。

输出：作业 J_i 的任务矩阵。

1) 初始化当前链路集合 E_{cur}
2) **while**(queue(J) is not **null**)
3) 　J_i = Deque(J)
4) 　S_i=Scheduling(J_i)
5) 　J_i.type = Prediction(S_i)
6) 　　**switch**(J_i.type)
7) 　　**case** C_0:
8) 　　　J_i.S = S_i
9) 　　endcase
10) 　　**case** C_1:
11) 　　　E_i = Sort(E_{cur})
12) 　　　E_{cur}= E_{cur}- E_i
13) 　　　update(J_i.S)
14) 　　　　if J_i is completed
15) 　　　　　E_{cur}= E_{cur}- E_i
16) 　　endcase
17) endwhile

算法 1 的输入是作业队列、调度器对该作业的预调度信息以及当前 SDN 链路集合；算法输出是作业 Map 任务的部署节点集合，即一个作业的 Map 任务在整个集群中的分布。提取作业队列中作业采用原调度算法产生预部署方案(行 4)，再使用神经网络预测模型判断作业的类型(行 5)。根据作业预测类型来判断(行 6)，如果是通信惰性作业，则维持原有部署方案(行 7-9)；如果是通信活跃作业，启动 SDN 控制器检测当前链路可用带宽，对当前链路集的可用带宽排序后提取出最优链路子集(行 11)，更新该作业部署节点集(行 13)；当该作业运行结束后回收 SDN 链路资源(行 14-15)。

4　仿真实验及分析

本节中使用 32 个节点的集群建立两个不同的实验床环境。第一种实验环境是通过三层交换机来模拟数据中心的树型拓扑结构，共分成 4 个机架，每个机架中部署 8 个计算节点。第二种实验环境部署软件定义网络环境。选择胖树拓扑网络，该网络中包含 50 台交换机和 32 台计算节点，控制器和交换机分别选用 Ryu 和 OpenSwitch。每个计算节点的硬件配置为 Intel Xeon E5620 2.4GHz 的双核 CPU，16GB DDR RAM 和 2TB 的 SATA 硬盘。软件系统使用 Ubuntu15.0，JDK1.8, Hadoop1.2.1 等。

首先，实验选择 Hibench 基准程序集中的 Sort,WordCount,TeraSort,Bayesian Classification 和 K-means Cluster 作为测试作业，表 1 中列出实验作业的数据规模和特征[18]。

表 1 测试作业特征

Table 1　Test job characteristics

作业名	简称	Map 输入	中间数据	Reduce 输出
Sort	st	120 GB	120 GB	120 GB
WordCount	wc	200 GB	11.23 GB	4.1 GB
TeraSort	ts	1TB	140 GB	1TB
Bayesian Classification	bc	78 GB	49 GB	43 GB
K-means Cluster	kc	66 GB	330KB	4.6KB

为了方便表述，文中数据混洗通信优化方案记为 SCOD(shuffle communication optimization deployment)，未优化的 MapReduce 方案直接采用 Hadoop 的默认调度方法，简记为 MR(MapReduce)。

4.1　通信性能对比及分析

由于 MapReduce 作业完成时间由计算时间和中间数据通信时间构成。为了减少实验过程的复杂性，忽略调度造成的计算偏差，设相同数据规模计算作业的计算时间相同，因此本节用计算作业最终完成时间来代替中间数据通信时间。一个相同数据规模作业运行在不同的实验床上，运行时间长的作业认为中间数据通信时间长。如图 4 所示。

基准作业 st,ts,bc 出现大约 4.7%~6.2%的性能提升，然而 wc 和 kc 未有明显的性能优化。由表 1 可见，中间数据量大的三个作业 st,ts,bc 产生了较好的通信优化效果；中间混洗数据量较小的作业 wc 和 kc 优化效果不明显。因此该通信优化方案对数据混洗密集作业的优化效果好。

为了进一步验证通信性能的优化，选择数据混洗密集型的 TeraSort 作业来进一步分析。当 Map 阶段结束后出现中间数据的混洗 Shuffle 过程，然后直到 Reduce 阶段结束。从技术角度可以监测到 Map 阶段的结束时刻和作业完成的时刻。如图 5 所示，计算作业数据规模分别为 32GB、64GB、128GB、

512GB 和 1TB，纵坐标表示数据混洗和 Reduce 阶段占整个作业计算时间的比例，比例相对减少则说明通信过程得到优化。随着数据规模的增加 SCOD 数据混洗和 Reduce 阶段所占总计算时间的比例逐渐减少，说明随着数据规模增加通信优化效果对计算性能的影响较明显，可见对大数据密集计算的性能提升具有较好意义。

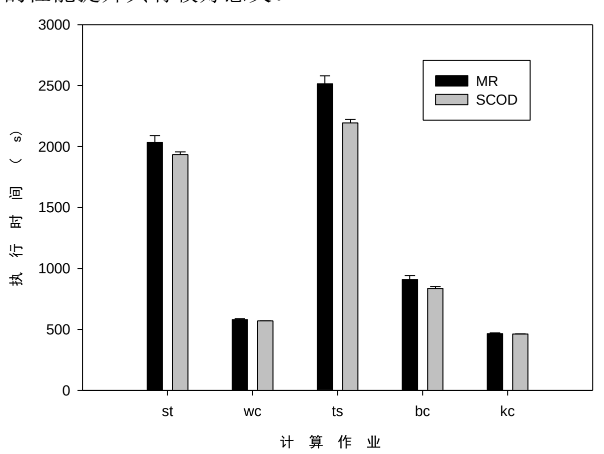


图 4　通信优化方案与基准方法性能比较

Fig. 4　Performance comparison between communication optimization schemes and benchmark methods

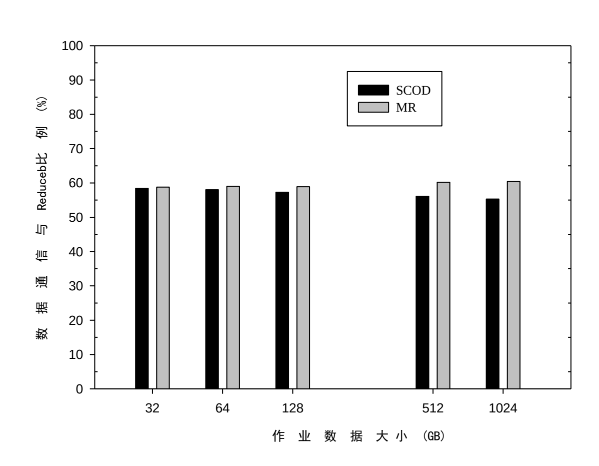


图 5　不同数据规模的 TeraSort 通信优化对比

Fig. 5　Terasort communication optimization comparison of different data scales

4.2　系统扩展性分析

随着计算节点的增加中间数据通信过程变的更加复杂，通信优化方案能否适应集群扩展成为一个重要问题，本节重点考虑系统扩展性对通信优化的影响。为了增加计算节点数量，在 SDN 的胖树结构上分别启动 32,64,128 个虚拟机节点来执行 TeraSort 作业，通过分析数据混洗阶段所占整个运行时间的比例来考察通信优化效果。如图 6 所示，随着计算节点数量的增加数据混洗所占比例逐渐下降，而且随着数据规模增加 Shuffle 所占比例也逐渐下降。实验表明当集群节点和数据规模增加后，该方案的优化效果明显增加。当集群节点数量较小时，网络背景流量掩盖了中间数据通信的优化效果，因此当数据规模小和网络规模小时优化效果不明显。

4.3　作业执行时间对比

为了提高实验验证效果，再从 PUMA[19]和 HiBench 基准程序集选了四个作业：sefl-join,inverted-index,pagerank 和 term-vector，这些作业的中间数据量大约在 32 ~300 GB 内，因此属于数据混洗密集型的作业。对于其他的作业如 histogram-movies,histogram-ratings 和 grep，由于数据交换较少而没有作为评估作业。本小节为了简化实验操作，如图 7 中 y 轴所示，直接比较作业的执行时间来分析通信优化效果，并以 Hadoop 中的调度模型作为基准，将作业的执行时间规范化为 1。

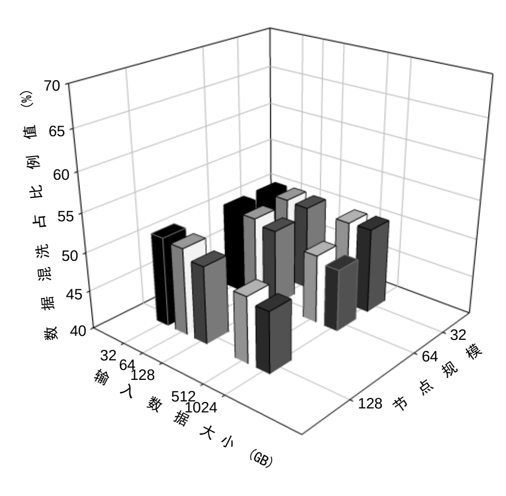


图 6　不同节点规模的 TeraSort 通信优化对比

Fig. 6　Terasort communication optimization comparison of different node scales

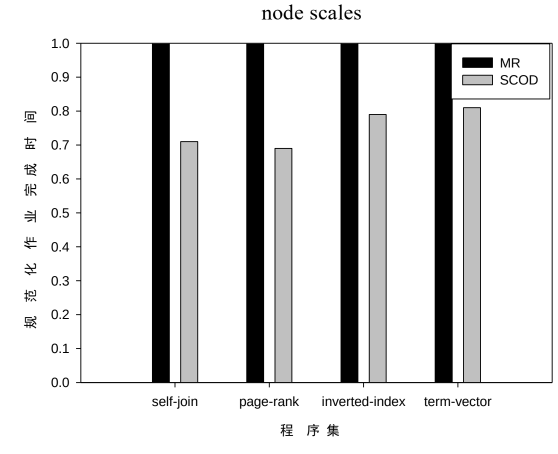


图 7　作业执行时间对比分析

Fig. 7　Contrastive analysis of job execution time

SCOD 在执行如图 7 的实验作业时，作业整体执行时间比 MR 基准调度模型要少。由于相同数据规模作业的 Map 和 Reduce 的计算时间基本相同，因此作业执行减少的时间为中间数据混洗的通信时间，进一步验证了所提方案的优化效果。虽然作业 inverted-index 和 term-vector 属于数据混洗密集型作业，由于计算中产生的中间数据量较小，通信优化效果相对较差；而作业 self-join 和 pagerank 的中间数据规模较大，通信效果明显好于前两种作业。

5　结束语

针对 MapReduce 计算模式中数据混洗密集型的作业，设计优化中间数据通信的调度方案。利用 BP 神经网络建立作业预测模型，判断出数据通信密集作业后再把该作业的任务部署到网络链路较好的计算节点中，实现以存储局部性来换取通信局部性的优化策略。为了适应各种网络拓扑结构采用 SDN 技术来寻找链路较优的节点，能够灵活地应对数据中心内各种网络结构的扩展需求，并且适应 MapReduce 模式的各种调度算法。实验表明能够有效控制中间数据混洗阶段的局部性，减少中间数据传输延迟和作业的整体执行时间，具有较好的通信优化效果。然而在大数据作业的实时计算中，由于作业动态提交而且网络链路具有时变性。未来深入研究

SDN 链路规划与作业动态调度的融合工作，建立适合复杂多租户应用场景的中间数据通信优化模型，进一步提高模型的应用价值。

参考文献：

[1] Zhang Jiaxing, Zhou Hucheng, *et al.* Optimizing data shuffling in data-parallel computation by understanding user-defined functions [C]// Proc of the 9th USENIX Conference on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2012: 295-308.

[2] Shadi I, Jin Hai, Lu Lu, *et al.* Leen: locality//fairness-aware key partitioning for mapreduce in the cloud [C]// Proc of the 2nd IEEE International Conference on Cloud Computing Technology & Science. Piscataway,NJ: IEEE Press, 2010: 17-24.

[3] Palanisamy B, Singh A, Liu Ling, *et al.* Purlieus: locality-aware resource allocation for mapreduce in a cloud [C]//Proc of International Conference for High Performance Computing, Networking, Storage and Analysis.New York: ACM Press, 2011: 58.

[4] Guo Yanfei, Rao Jia, *et al.* iShuffle: improving hadoop performance with shuffle-on-write [J]. IEEE Trans on Parallel and Distributed System, 2017, 28 (6): 1649-1661.

[5] 陆菲菲，郭得科，方兴等．数据中心网络高效数据汇聚传输算法 [J]. 计算机学报，2016，39 (9)：1750-1762. (Lu Feifei, Guo Deke, Fang Xing, *et al.* Efficient data aggregation transfers in data center networks [J]. Chinese Journal of Computers, 2016, 39 (9): 1750-1762.)

[6] Ke Huan, Li Peng, *et al.* On traffic-aware partition and aggregation in mapreduce for big data applications [J]. IEEE Trans on Parallel and Distributed System, 2016, 27 (3): 818-828.

[7] Rahman M W, Islam N S, *et al.* A comprehensive study of MapReduce over lustre for intermediate data placement and shuffle strategies on HPC clusters [J]. IEEE Trans on Parallel and Distributed System, 2017, 28 (3): 633-646.

[8] Wang Jihe, Wang Danghui, *et al.* Similarity-based node distance exploring and locality-aware shuffle optimization for Hadoop MapReduce [C]// Proc of IEEE International Conference on Smart Cloud. Piscataway,NJ: IEEE, 2017: 103-108.

[9] Ye Yu, Qian Chen. Space Shuffle: a scalable, flexible, and high-performance data center network [J]. IEEE Trans on Parallel and Distributed System, 2016, 27 (11): 3351-3365.

[10] Wang Jihe, Wang Danghui, *et al.* A locality-aware shuffle optimization on fat-tree data centers [J]. Future Generation Computer Systems, 2018, 89: 31-43.

[11] Narayan S, Bailey S, Daga A. Hadoop acceleration in an openflow-based cluster [C]// Proc. of SC Companion: High Performance Computing, Networking Storage and Analysis.Piscataway,NJ: IEEE Press, 2012: 535-538.

[12] Qin Peng, Dai Bin, *et al.* Bandwidth-aware scheduling with SDN in Hadoop: a new trend for big data [J]. IEEE Systems Journal, 2017, 11 (4): 2337-2344.

[13] 杨军，吕璐,徐冠,等．基于 SDN 的 MapReduce 带宽优化设计 [J]. 计算机应用研究，2016，33 (10)：3109-3113. (Yang Jun, Lyu Lu,Xu Guan, *et al.* Optimization design of MapReduce based on SDN [J]. Journal of Application Research of Computers, 2016, 33 (10): 3109-3113.)

[14] Khaleel A, Al-Raweshidy H. Optimization of computing and networking resources of a Hadoop cluster based on software defined network [J]. IEEE Access, 2018, 6: 61351-61365.

[15] Zhang Fan, Sakr M, *et al.* Empirical discovery of power-law distribution in mapreduce scalability [J]. IEEE Trans on Cloud Computing, 2017, PP (99): 1.

[16] Zhang Fan, Cao Xibin, Zou Jingxiang. Cost estimating for small satellite using fuzzy BP neural networks [J]. Systems Engineering and Electronics, 2000, 22 (10): 75-78.

[17] Qi Chao, Wu Jiangxing, *et al.* An intensive security architecture with multi-controller for SDN [C]// Proc of IEEE Conference on Computer Communications . Piscataway,NJ: IEEE, 2016: 401-402.

[18] Huang Shengsheng, Huang Jie, *et al.* The Hibench benchmark suite: characterization of the mapreduce-based data analysis [C]// Proc of the 26th IEEE International Conference on Data Engineering. Piscataway,NJ: IEEE, 2010, 74: 41-51.

[19] PUMA. Purdue MapReduce benchmark suite [EB/OL]. [2012]. http://web. ics. purdue. edu/fahmad/benchmarks. htm.